Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matija Obid

Digitalni osciloskop

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: prof. dr. Patricio Bulić

Ljubljana, 2016

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Implementirajte preprost enokanalni digitalni osciloskop na mikrokrmilniku STM32F407. Osciloskop naj ima osnovne načine proženja (na nivo, na prehod, ipd.). Osciloskop naj bo preko povezave USB povezan z osebnim računalnikom, na katerm implementirajte grafični vmesnik za prikazovanje merjenih signalov in upravljanje z osciloskopom.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Matija Obid sem avtor diplomskega dela z naslovom:

Digitalni osciloskop

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Patricia Bulića,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, 2. septembra 2016 Podpis avtorja:

Za končano diplomsko delo se zahvaljujem mentorju Patriciu Buliću ter vsem, ki so mi pomagali pri izdelavi diplomskega dela.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
CRO	Cathode ray oscilloscope	osciloskop s katodno cevjo
DSO	Digital storage oscilloscope	digitalni spominski osciloskop
DPO	Digital phosphor oscilloscope	digitalni fosforni osciloskop
ADC	Analog to digital converter	analogno-digitalni pretvornik
DAC	Digital to analog converter	digitalno-analogni pretvornik
JTAG	Join test action group	(ni prevoda)
SWD	Serial wire debugging	razhroščevanje po eni žici
USB	Universal serial bus	univerzalno zaporedno vodilo
GCC	GNU compiler collection	GNU zbirka prevajalnikov
GNU	GNU is not unix	GNU ni UNIX
DIV	Division	razdelek
GPIO	General purpose input/output	splošnonamenski vhod/izhod
DMA	Direct memory access	neposredni dostop do pomnilnika
OCD	On-Chip debugger	mikroprocesorski razhroščevalnik
GDB	GNU debugger	GNU razhroščevalnik
PWM	Pulse-width modulation	pulzno-širinska modulacija
LSB	Least significant bit	najmanj pomemben bit
MSB	Most significant bit	najbolj pomemben bit
AWD	Analog watchdog	analogni kuža-pazi
RAM	Random access memory	bralno-pisalni pomnilnik

Povzetek

Osciloskop je nepogrešljiva naprava v elektrotehniki. Z njo lahko opazujemo električni signal v realnem času in tako najdemo morebitne napake v delovanju električnega vezja.

V diplomski nalogi je najprej predstavljeno delovanje analognega ter digitalnega osciloskopa. Predstavljena je osnovna terminologija, ki jo srečamo pri osciloskopih ter procesi, ki jih digitalni osciloskop izvaja.

V drugem delu diplome izdelamo enostaven digitalni osciloskop, ki omogoča zajem signala, proženje ob različnih digitalnih dogodkih ter prikaz signala. V ta namen uporabimo razvojno ploščico STM32F4-Discovery, ki jo dobimo razmeroma poceni. Z njo zajamemo signal, ga preko vodila USB prenesemo na računalnik ter tam prikažemo. Da osciloskop testiramo, zajamemo signala PWM ter RS232.

Ključne besede: osciloskop, signali, vzorčenje, proženje, servomotor, signal RS232.

Abstract

An oscilloscope is a fundamental device in electrical engineering. With an oscilloscope we can view the electrical signal in real time to help us find possible glitches in the electrical circuit.

This dissertation starts off with a presentation of what an oscilloscope is. The basic terminology of oscilloscopes is introduced in this part along with an explanation of how an oscilloscope works.

In the second part we create a simple digital oscilloscope, which can capture the signal and display it to the screen of a PC. For this purpose, we use a development board STM32F4-Discovery, which can be acquired at a reasonable price. We use it to detect a digital event on the signal and then capture it. Captured samples are transferred to a PC using USB protocol and are displayed to the screen. We test our digital oscilloscope by capturing PWM and RS232 signals.

Keywords: oscilloscope, signals, sampling, triggering servomotor, RS232 signal.

1. Uvod

Osciloskop je naprava, ki se uporablja za prikaz električnega signala. Z njim lahko zaznamo spremembe signala, ki se zgodijo v zelo kratkem času. Opazujemo lahko tudi druge veličine, naprimer zvok, vendar moramo za to imeti pretvornik, ki nam dano veličino spremeni v električni signal. Za opazovanje zvoka bi potrebovali mikrofon.

Nekoč so bili osciloskopi drage in velike naprave, ki smo jih srečali v laboratorijih. Ker je bilo njihovo delovanje analogno, se signal ni nikamor shranil, zato smo z njimi lahko opazovali le signal v realnem času.

Z napredkom digitalizacije je možno napetost signala pretvoriti v digitalno vrednost – vzorec. Tako lahko na signal gledamo kot na vrsto oziroma zaporedje vrednosti, ki predstavljajo napetost signala v danem časovnem intervalu. To nam prinaša veliko prednosti, saj lahko signal (zaporedje vzorcev) shranimo za kasnejšo analizo ali obdelavo. Shranjen signal lahko tudi analiziramo s pomočjo računalniških programov ali ga natistnemo. Lahko ga tudi popravimo in z ustreznim generatorjem električne napetosti ponovimo/rekonstruiramo.

Po velikosti so digitalni osciloskopi manjši kot analogni. Poznamo tudi žepne osciloskope, žepne velikosti, ki nam ponujajo vse osnovne zmogljivosti osciloskopa.

V diplomski nalogi bom predstavil delovanje analognega in digitalnega osciloskopa, ter kako lahko s poceni razvojno ploščico zajamemo signal, ga prenesemo na računalnik ter tam prikažemo za nadaljnjo analizo.

Razvojno ploščico STM32F4-Discovery bomo uporabili za digitalizacijo signala. Ko bo signal enkrat pretvorjen v digitalne vzorce, bomo te preko vodila USB prenesli na računalnik. Na računalniku bo programska oprema, razvita v jeziku

python, sprejela vzorce in jih prikazala na grafu napetosti v odvisnosti od časa.

V diplomski nalogi se ne bomo spuščali v analizo signalov, bomo pa z osciloskopom zajeli signal RS232 ter signal PWM (pulzno-širinska modulacija). Prvi signal se uporablja za serijsko povezavo med napravami, drugega pa srečamo pri elektroniki, ki krmili radijsko vodene modele.

2. Osciloskop

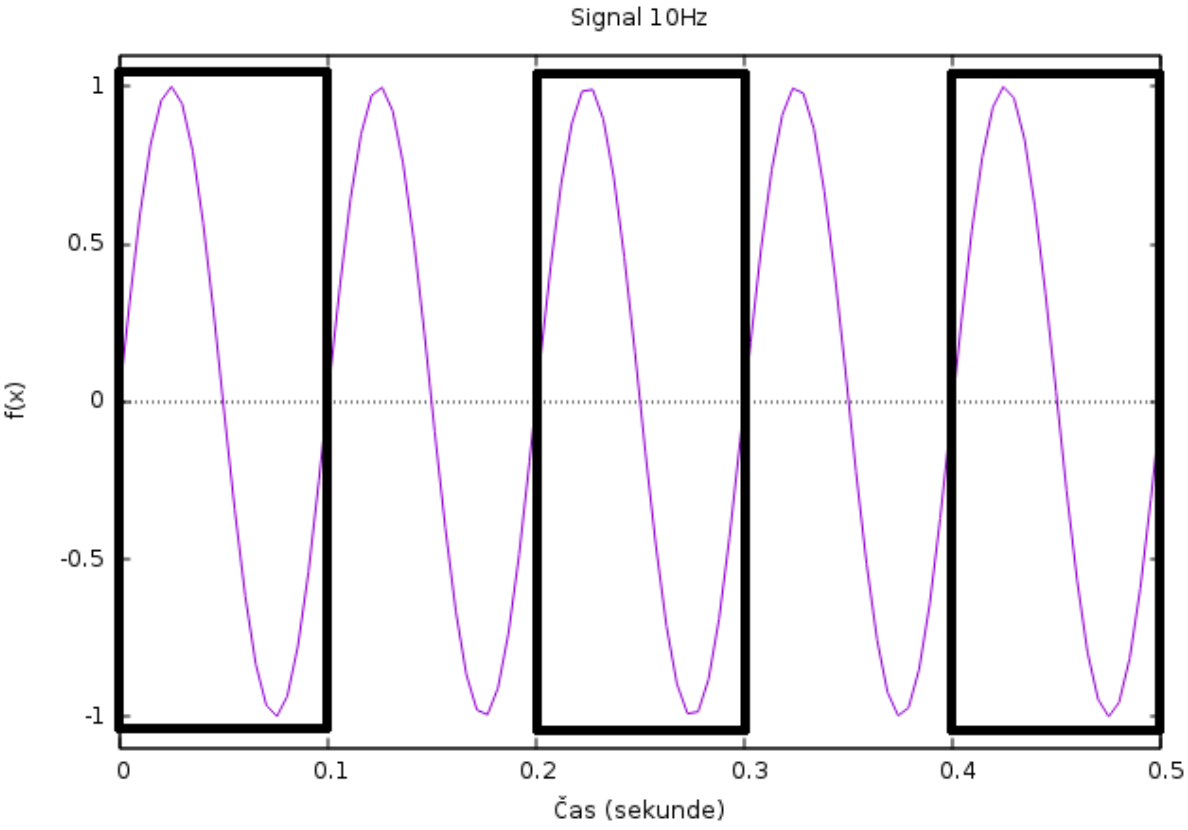
2.1 Kaj je osciloskop

Osciloskop je merilna naprava, ki nam omogoča opazovanje električnega signala v časovnem prostoru. To pomeni, da nam na zaslonu prikazuje graf signala, pri katerem os `x` predstavlja časovno bazo oziroma časovno okno, v katerem je bil signal zajet, os `y` pa predstavlja merilno območje oziroma napetost vhodnega signala. Nekateri osciloskopi imajo s pomočjo intenzitete prikazano še tretjo dimenzijo `z`. Z njo lahko opazujemo, kako se signal ponavlja skozi čas.

Z osciloskopom lahko spremljamo delovanje vezja ter poiščemo morebitne težave, ki jih sicer ne bi videli. Točko zajema signala na vezju določimo s konico sonde. Sonda nam pomaga, da želen signal z vezja enostavno pripeljemo do osciloskopa ter da se pri prenosu ta čim manj popači.

Ker ima tudi osciloskop svojo notranjo upornost, z merjenjem signala obremenimo njegov izvor, kar pripelje do tega, da se signal oslabi. Zato imajo sonde z oznako x10 možnost, da napetost signala znižajo v merilu $1 : 10$, ter tako dosežejo 10-krat manjšo obremenitev signala. Lahko pa na ta način merimo tudi signal, ki ima do 10-krat višjo napetost.

Ker se signali običajno spreminjajo prehitro, da bi jih opazovali v realnem času, se omejimo na opazovanje njihove periode, dela periode ali več zaporednih period. Le na tak način dosežemo, da imamo na zaslonu statično sliko, iz katere lahko tudi kaj odčitamo.



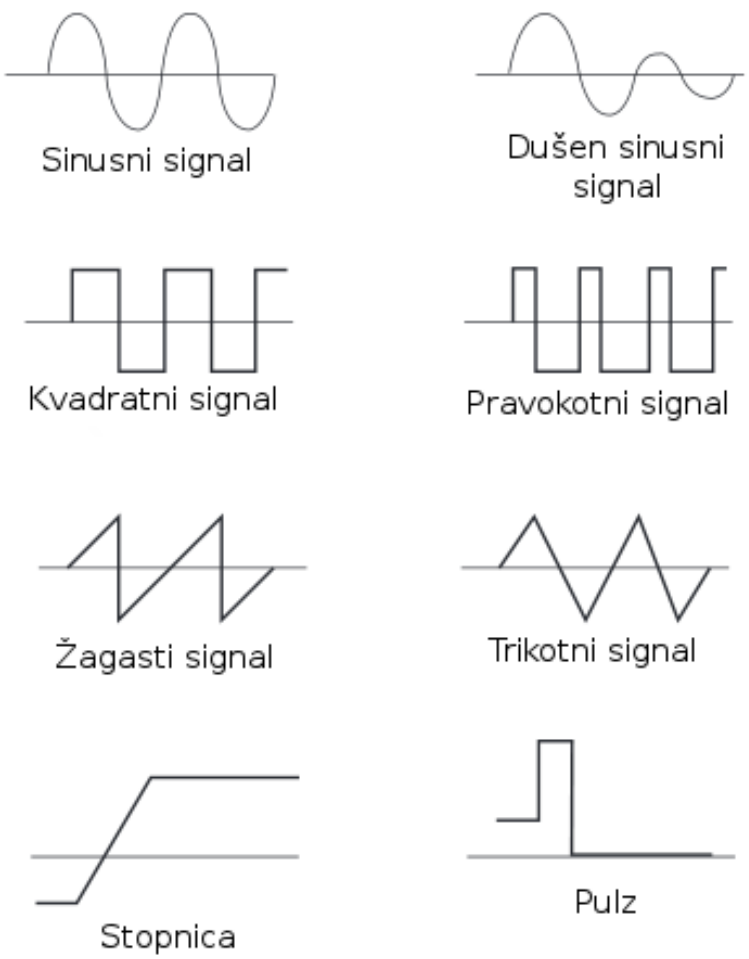
Slika 2.1: Prikaz na osciloskopu[1]

Na sliki 2.1 je prikazan sinusni signal v realnem času. Z okvirji na sliki so prikazani deli signala, ki jih prikazuje osciloskop. V času med okvirji osciloskop obdeluje zajete vzorce za prikaz.

V primeru, da signal ni periodičen, se osredotočimo na posamezen dogodek na signalu.

Dogodek signala je vsaka sprememba napetosti signala. Če se ta ponavlja, se imenuje periodičen dogodek. Glede na to, iz kakšnih dogodkov je sestavljen signal, ločimo več vrst osnovnih signalov. Na sliki 2.2 so prikazani sinusni, pravokotni, žagasti, trikotni, stopničasti signal ter pravokotni pulz.

[1] Med posameznimi prikazi ni nujno, da preteče natanko ena perioda. Lahko jih je tudi več.

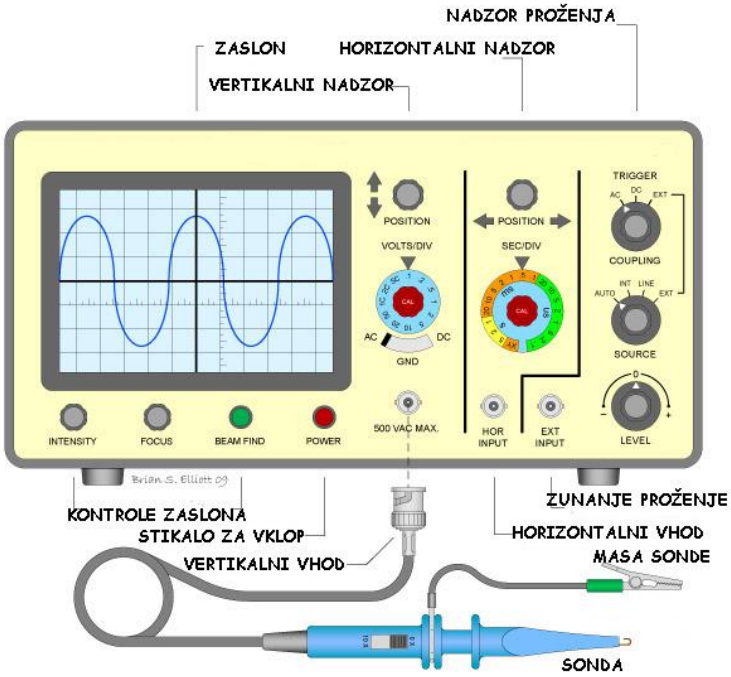


Slika 2.2: Pogostejše oblike signalov [31, str. 8]

Sinusni in pravokotni signal bomo merili v poglavju z meritvami. Žagasti signal bomo srečali pri delovanju analognega osciloskopa. Stopničasti signal oziroma dogodek pa si bomo podrobneje ogledali pri prožilcih.

2.2 Opis osciloskopa

Da bi lažje razumeli, kako deluje osciloskop, si najprej oglejmo, kako izgleda in kako se uporablja.



Slika 2.3: Osnoven osciloskop [19]

Na sliki 2.3 je prikazana skica osnovnega osciloskopa. Njegovi glavni deli so:

- sonda za zajem signala,
- kontrolni gumbi,
- zaslon za prikaz sledi signala,
- eden ali več vhodov, na katere priključimo sondo za zajem signala.

Signal, ki ga opazujemo, se nam prikazuje na zaslonu. Osciloskop nam signal

vzorči s frekvenco, ki je višja od frekvence opazovanega signala,[2] ter nam vzorce prikazuje na zaslon. Če bi na zaslonu prikazovali signal v realnem času, bi se le ta lahko tako hitro spreminjal, da z zaslona ne bi mogli odčitati ničesar. Zato se raje omejimo ali na en dogodek signala ali na periodične signale. Kaj je perioda ali dogodek signala in kje se ta začne, določimo s prožilcem.

2.2.1 Zaslon

Na zaslonu se nam prikazuje `sled signala` (`sweep`). Ta je sestavljena iz digitalnih vzorcev, ki jih analogno-digitalni pretvornik `ADC` (Analog to digital converter) pretvori iz napetosti signala v vsakem ciklu zajemanja. Vsak vzorec nam pove, kakšna je bila napetost signala v trenutku zajema. V vertikalni smeri imamo prikazano napetost, v horizontalni smeri pa čas zajema. Za lažje odčitavanje veličin z zaslona je ta z mrežo razdeljen na kvadratke, ki določajo merilo signala. Z njimi si pomagamo, da z zaslona odčitamo čas in napetost.

2.2.2 Kontrole za časovno bazo in napetostno območje

S kontrolami na zaslonu prikažemo del signala, ki ga želimo opazovati.

Kontrole za časovno bazo in napetostno območje so si med seboj podobne. Z njimi upravljamo pomik in povečavo signala v želeni smeri. S povečavo signala v navpični smeri nastavljamo napetost, ki jo predstavlja en razdelek mreže na zaslonu. S povečavo signala v vodoravni smeri nastavljamo časovno bazo osciloskopa, oziroma kolikšna časovna enota je prikazana z enim razdelkom na zaslonu. S pomikom signala lahko signal poravnamo z mrežo na zaslonu, da lažje odčitamo želeno veličino.

2.2.3 Proženje

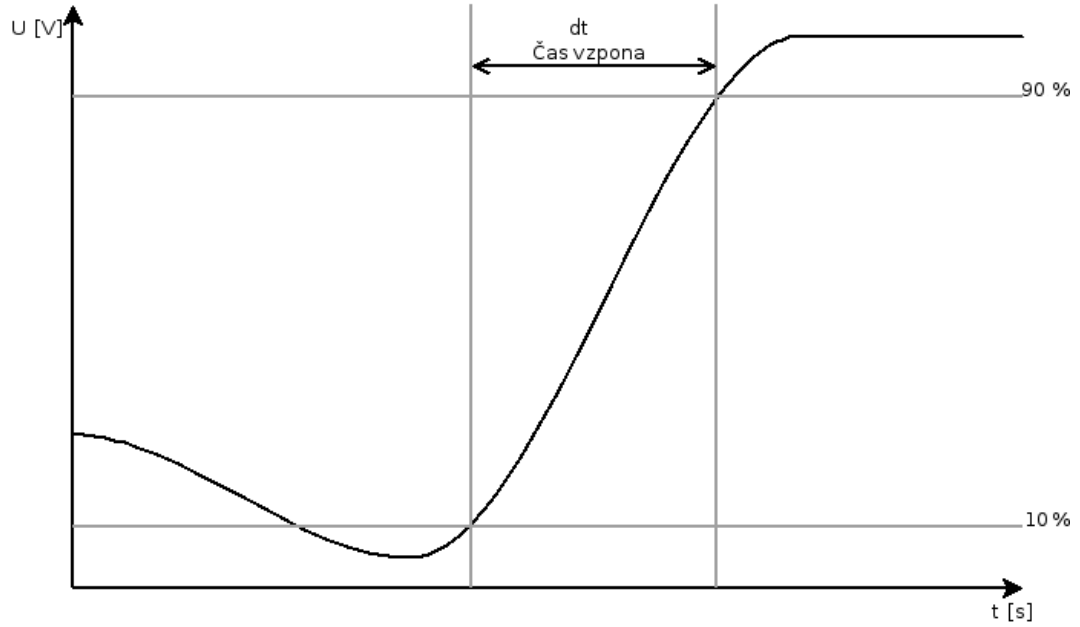
Prožilci so pomemben del osciloskopa, saj nam pomagajo umiriti sliko in na zaslon prikazati del signala, ki nas zanima. Proženje se zgodi, ko osciloskop na signalu

[2] S posebno metodo lahko merimo tudi signale, ki imajo višjo frekvenco od frekvence vzorčenja. Več o tem v poglavju 2.4.

zazna določen dogodek.

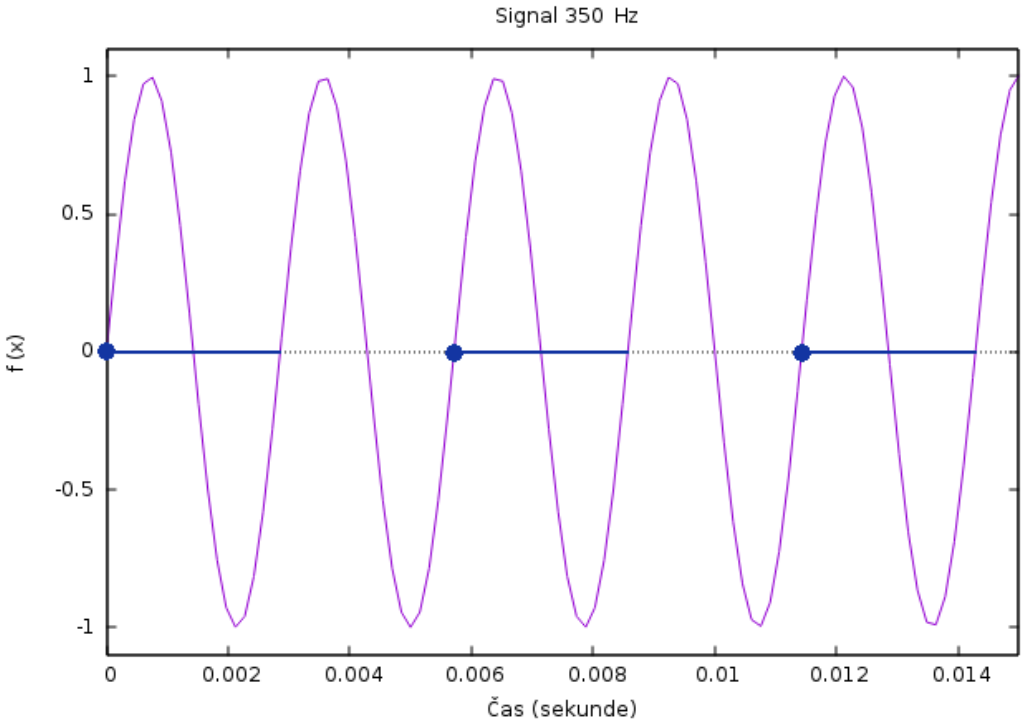
Prožilec se lahko sproži na več različnih dogodkov, ki so prikazani na sliki 2.2.

- Ko signal doseže določeno napetost. Nastavimo lahko prag vhodne napetosti, ki loči nizko in visoko stanje, ter v katerem stanju se bo prožilec sprožil.
- Ob prehodu signala iz nizkega v visoko stanje ali obratno. Nastavimo lahko prag vhodne napetosti in čas prehoda, ob katerem se bo prožilec sprožil. Prožilec se sproži, kadar je čas vzpona (slika 2.4) ali padca signala enak/ manjši/večji od časa, ki ga vnaprej določimo pri prožilcu.
- Ob pulzu signala. Prožilec se sproži, če je širina pulza enaka/manjša/večja od nastavljenega časa. Nastavimo lahko tudi prag vhodne napetosti, ki ga mora pulz preseči.



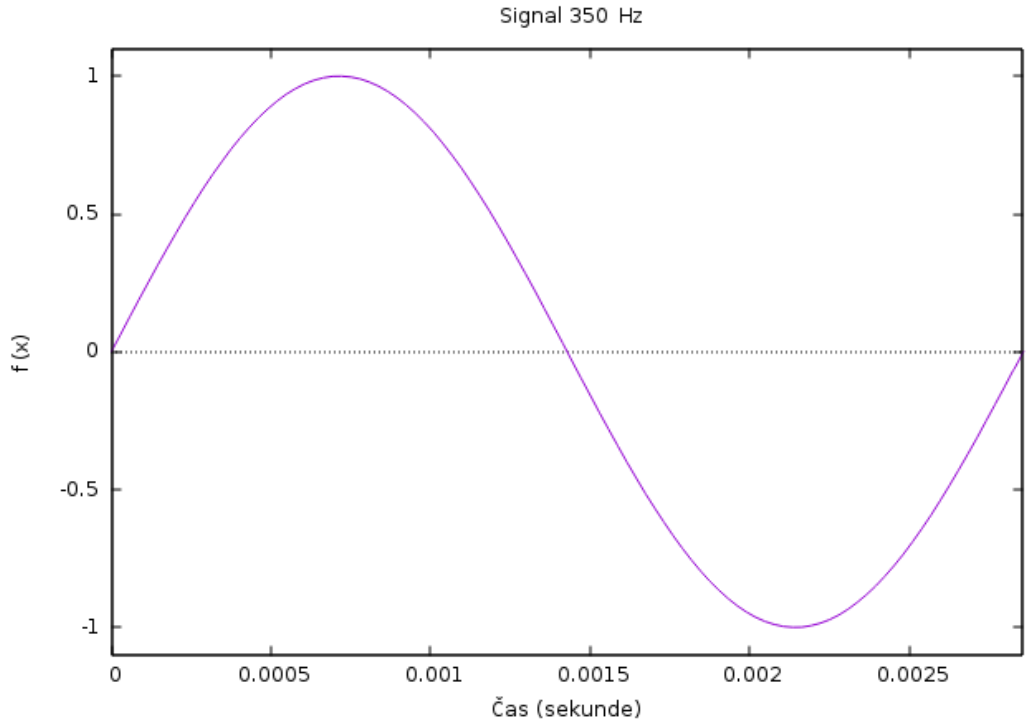
Slika 2.4: Čas vzpona signala

Slika 2.5 prikazuje signal, ki ga opazujemo s pomočjo prožilca. Modre pike (v tisku samo pike) ponazarjajo točko proženja. Proženje se zgodi ko signal preide



Slika 2.5: Proženje ob prehodu iz nizkega v visoko stanje

iz nizkega v visoko stanje, pri čemer je meja 0 V. Ko se prožilec aktivira, začne osciloskop z zajemom sledi signala. Ta čas je na sliki označen z modro črto (v tisku odebeljena črna črta na `x` osi). Po zajemu osciloskop obdela ter prikaže zajete vzorce, nato se spet vrne v stanje, v katerem čaka na proženje prožilca. V tem primeru bi na osciloskopu videli sled signala, ki je prikazana na sliki 2.6.



Slika 2.6: Proženje ob prehodu iz nizkega v visoko stanje

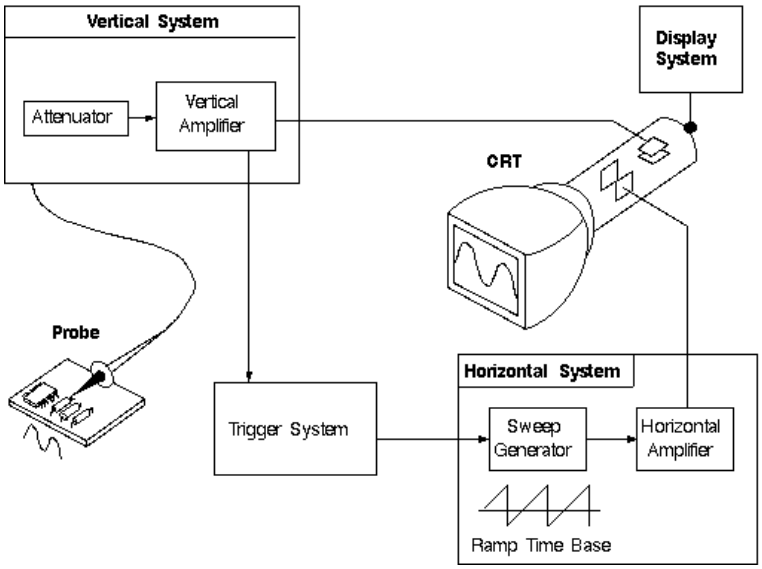
2.3 Delovanje osciloskopa

Glede na delovanje ločimo dve vrsti osciloskopov:

- analogni oz. katodni osciloskopi (`CRO`)
- digitalni spominski osciloskopi (`DSO`)

2.3.1 Analogni osciloskop

Analogni osciloskop uporablja za prikaz katodni zaslon, ki ga krmilita dve enoti. Prva enota krmili premik snopa elektronov v vodoravni smeri. Kadar enota zazna na signalu določen dogodek, se začne snop elektronov premikati iz leve strani proti desni. To povzroči, da se na zaslonu prikaže sled signala. Od hitrosti premika je odvisno, kakšen čas bo prikazan v razdelku na zaslonu. Hitrejši kot je premik, krajši čas je prikazan na zaslonu, kar pomeni, da opazujemo hitrejše signale. Druga enota krmili premik snopa v navpični smeri glede na napetost signala.



Slika 2.7: Diagram delovanja analognega osciloskopa [1]

Na sliki 2.7 je prikazano delovanje analognega osciloskopa. Signal je najprej speljan skozi ojačevalnik ali slabilnik, kjer se ustrezno ojača oziroma oslabi. Ko-

liko se bo signal ojačal ali oslabil, nastavimo s kontrolami, s katerimi krmilimo napetostno območje signala.

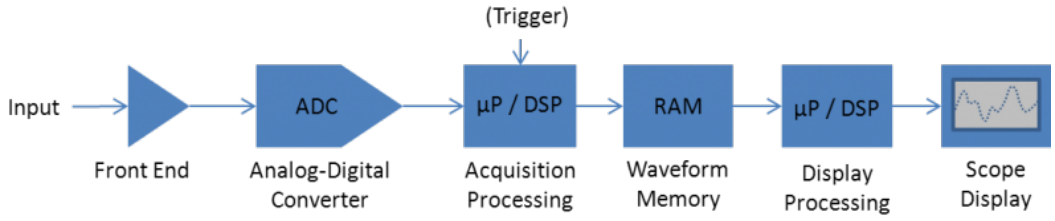
Nato se signal razdeli in gre v dve podenoti. Kot smo že omenili, prva enota vsebuje prožilec, ki na signalu zazna dogodek. Ko ga zazna, generira žagasti signal, ki povzroči premik snopa signala v vodoravni smeri.

Druga enota vsebuje zakasnilno linijo, ki signal zadrži za določen čas ter z njim usmerja snop eletronov v navpični smeri.

Bolj pogosto kot snop elektronov zadene isto točko na zaslonu, bolj se ta zasveti. To omogoča plast fosforja, s katero je premazan zaslon.

2.3.2 Digitalni osciloskop

Digitalni osciloskop se od analognega razlikuje po tem, da signal ne krmili snopa elektronov, ampak se napetost signala pretvori v vzorec, ki se nato shrani v pomnilnik. To prinese veliko prednosti, saj signal, ko je enkrat shranjen, lahko obdelujemo, analiziramo, prikažemo ali natisnemo. Za prikaz se uporablja točkovni zaslon, katerega slabost je pomanjkanje Z-koordinate. To slabost odpravlja digitalni fosforni osciloskop, a o njem več v poglavju 2.4.



Slika 2.8: Diagram delovanja digitalnega osciloskopa [2]

Delovanje digitalnega osciloskopa je prikazano na sliki 2.8. Signal je enako kot pri analognem osciloskopu najprej speljan skozi ojačevalnik ali slabilnik, kjer se ustrezno ojača oziroma oslabi. Drugi korak je `vzorčenje` signala. V tem koraku `ADC` digitalizira signal tako, da napetost signala na vhodu pretvori v digitalno vrednost. Tako iz napetosti signala pridobimo vzorce, ki so med seboj razmaknjeni za čas Δt. Čas med vzorci je enak obratni vrednosti frekvence ure ($\frac{1}{f_{adc}}$), ki krmili

`ADC`. Ta narekuje, koliko vzorcev bo ADC pretvoril v eni sekundi. Ta podatek se imenuje `frekvenca vzorčenja` (`sample rate`).

Naslednja stopnja je mikrokrmilnik, ki preverja zajete vzorce, dokler iz vzorcev ne zazna želenega dogodka. Ko ga zazna, začne s prenosom vzorcev v pomnilnik. Velikost pomnilnika, ki ga potrebujemo, je odvisna od dveh parametrov: želene frekvence vzorčenja in želene dolžine zajema. Višja kot je frekvenca vzorčenja, več bo vzorcev in krajši signal bomo lahko shranili v pomnilnik. Če želimo zajeti daljši signal, moramo zmanjšati frekvenco vzorčenja. To lahko naredimo tako, da zmanjšamo hitrost, s katero `ADC` vzorči signal. Lahko pa zmanjšamo število vzorcev, s katerimi je predstavljen signal. Več o tem v poglavju 2.3.3.

Če zajemamo hitre signale, se lahko zgodi, da je zajetih vzorcev manj, kot jih potrebujemo, da sestavimo valovno obliko. V tem primeru pride v poštev interpolacija. To je način, da z dodajanjem vmesnih točk pridobimo krivuljo. Vrste interpolacij so linearna, polinomska in bolj pogosta interpolacija s $\frac{sin(x)}{x}$ [31, str. 24].

Ko imamo pripravljene točke krivulje, jih lahko mikrokrmilnik obdela in shrani v grafični pomnilnik (display memory). To je pomnilnik, ki ima celico za vsako točko na zaslonu. Pri shranjevanju v grafični pomnilnik spet naletimo na isto težavo. Število vzorcev, ki jih želimo prikazati, je lahko preveliko, da bi vse shranili v grafični pomnilnik. To težavo lahko rešimo tako, da vsak vzorec posebej prenesemo v grafični pomnilnik [17]. Pri tem se nekatere točke prekrivajo. Prekrivanje točk ponazorimo z intenziteto točk. Več točk, kot se prekriva, večja bo intenziteta te točke. Poznamo pa tudi načine, kako zmanjšati število vzorcev. Ti so opisani v poglavju 2.3.3.

Če uporabnik pogled na signal približuje, se bo prekrivalo vedno manj točk, dokler ne bo prikazana ena točka na eno točko v grafičnem pomnilniku. Če bo uporabnik še nadalje povečeval prikaz, bo točk zmanjkovalo in vmesne točke bo potrebno pridobiti z interpolacijo.

2.3.3 Manjšanje števila vzorcev v časovnem intervalu

Če bi radi vzorčen signal prikazali na zaslonu, imamo problem, saj je vzorcev lahko veliko več, kot je točk na zaslonu. V tem primeru moramo iz N zajetih vzorcev pridobiti novo manjše število vzorcev N_n, ki se ujema s številom točk na zaslonu. Novopridobljeni vzorci signala morajo še vedno predstavljati isti signal, čas med njimi Δt pa se bo povečal na Δt_n.

Veljati mora: $N * \Delta t = N_n * \Delta t_n = T$, pri čemer je T trajanje celotnega signala.

Pri manjšanju števila vzorcev moramo iz več vzorcev pridobiti enega samega. To lahko naredimo na več načinov:

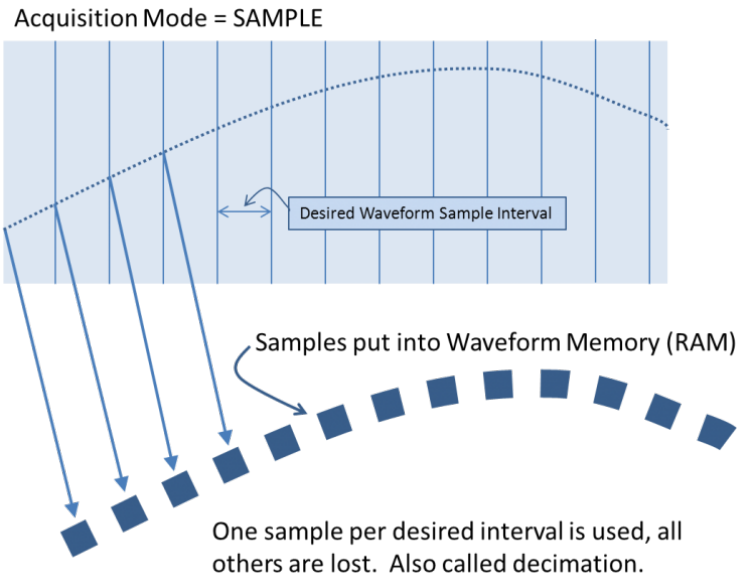
- Vzamemo prvi vzorec v intervalu (`sample mode`). Na ta način lahko izgubimo nam zanimive vzorce.
- Vzamemo največji in najmanjši vzorec v intervalu (`peak detect mode`). Tako ne bomo izgubili zanimivih vzorcev in bomo zaznali vsako nenadno spremembo signala.
- Vzamemo povprečje vseh vzorcev v intervalu (`HiRes mode`). Na ta način bomo iz signala odstranili šum in nenadne spremembe signala.

Za načina `HiRes mode` in `peak detect mode` obstajata še dve varianti (`envelope mode` in `average mode`), opisani v podpoglavju 2.3.3. Ti varianti delujeta na isti način, le da upoštevata vzorce iz več zaporednih zajemov. Delujeta le napri periodičnih signalih.

Sample mode

Je najenostavnejši način redčenja števila vzorcev. En vzorec iz intervala se vzame za sestavo novega signala, ostali vzorci se zavržejo. Način je enostaven in hiter, vendar lahko izgubimo nam zanimive vzorce.

Na sliki 2.9 so prikazani vzorci, ki jih zajame `ADC` v enem zajemu. Večji vzorci so zredčeni vzorci. Vzorce najenostavneje zredčimo tako, da iz intervala vzorcev vzamemo prvi vzorec. Intervali so na sliki med seboj ločeni z navpičnimi črtami.



Slika 2.9: Primer podaljševanja časa Δt med vzorci [16]

Peak detect mode

Ta način je namenjen odkrivanju nam zanimivih vzorcev, ki jih lahko izgubimo s prej omenjenim načinom.

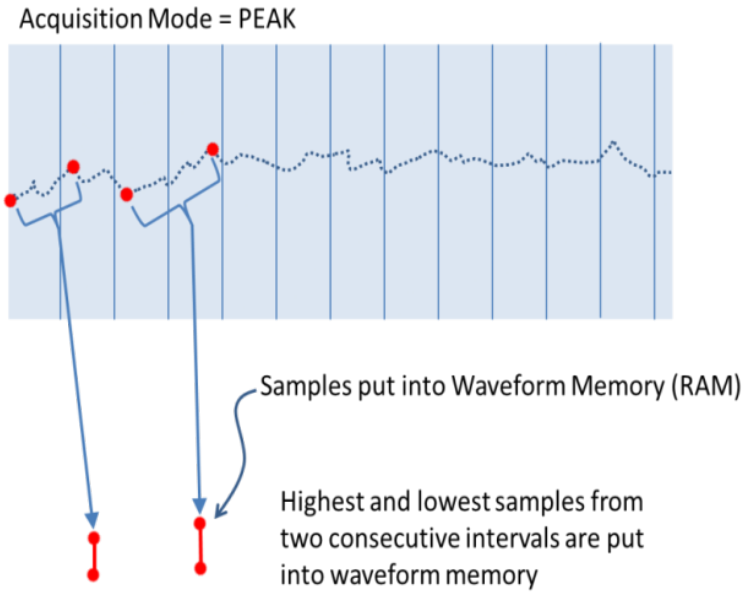
V tem primeru shranimo par vzorcev z najnižjo in najvišjo napetostjo iz dveh sosednjih intervalov (prikazano na sliki 2.10). Na ta način ne izgubimo nobenega vzorca z nenormalno visoko ali nizko napetostjo.

HiRes mode

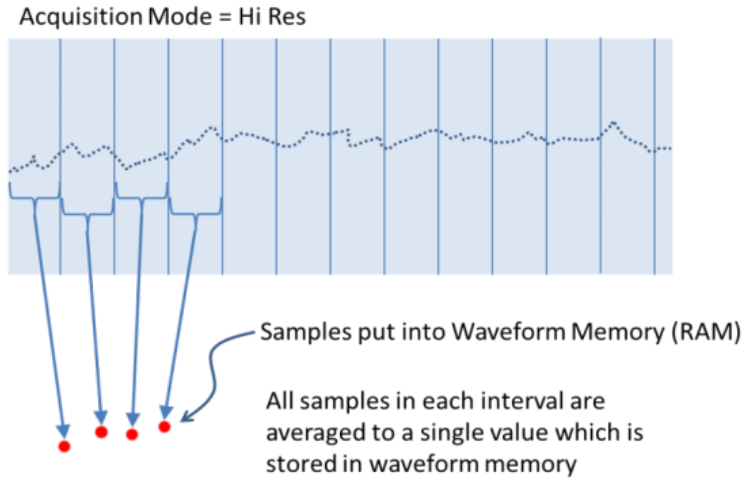
V visokoločljivostnem načinu (`High Resolution`) vzamemo povprečje vseh vzorcev iz intervala (prikazano na sliki 2.11). S tem zmanjšamo šum in nenavadne pojave na signalu. Ker vzamemo povprečje, se nam poveča tudi ločljivost novopridobljenih vzorcev. Če Δt vzorcev dvakrat podaljšamo, pridobimo 0,5 bita ločljivosti.

Več zaporednih zajemov

Zgornji načini delujejo na vsakem zajemu podatkov posebej. Obstajajo tudi načini, ki obdelujejo vzorce iz več zaporednih zajemov. Seveda delujejo samo pri peri-



Slika 2.10: Primer podaljševanja časa Δt med vzorci z načinom `peak` [12]



Slika 2.11: Primer podaljševanja časa Δt med vzorci z načinom HiRes [5]

odičnih signalih [31, str. 22]. Ti načini se po delovanju ne razlikujejo dosti od zgoraj omenjenih načinov.

Povprečenje (Average mode): enako kot pri `HiRes` načinu vzame povprečje vzorcev, le da pri tem načinu upoštevamo vzorce iz več zaporednih zajemov. Število zaporednih zajemov lahko nastavi uporabnik.

Ovojnica signala (Envelope mode): enako kot pri `peak mode` vzamemo par vzorcev z najnižjo in najvišjo napetostjo, le da upoštevamo tudi vzorce iz več zaporednih zajemov.

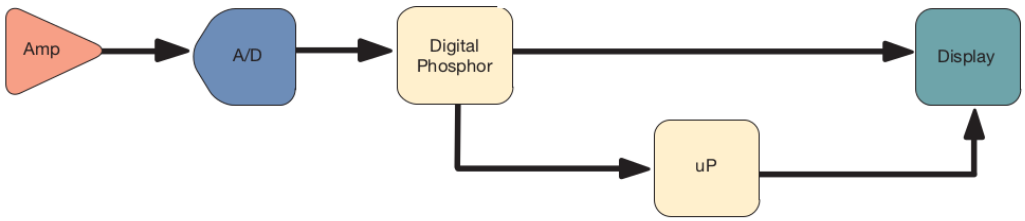
2.4 Vrste osciloskopov

Glede na delovanje in posebnosti, ki jih imajo nekateri osciloskopi, lahko ločimo naslednje vrste:

- digitalni fosforni osciloskopi (`DPO`),
- digitalni vzorčni osciloskopi,
- ročni osciloskopi,
- računalniški osciloskopi.

Digitalni fosforni osciloskop je podvrsta digitalnega osciloskopa, združuje dobre lastnosti analognega osciloskopa. S pomočjo paralelnega vezja doseže prikaz Z-koordinate (intenzitete), brez da bi se pri tem upočasnil proces zajema in obdelave signala. Intenziteta pri prikazu nam omogoča, da ne spregledamo napake, ki se pojavlja poredko, na naprimer vsakih 100 zajetih krivulj.

Za prikaz signala se uporablja običajen točkovni zaslon in ne fosforni, kot ga imajo analogni osciloskopi. Fosfor v imenu se nanaša na digitalni fosfor [31, str. 16], ki se uporablja za izdelavo vezij. V tem primeru se nanaša na paralelno vezje, ki služi kot baza (`fosforna digitalna baza`) za hranjenje intenzitete signala.



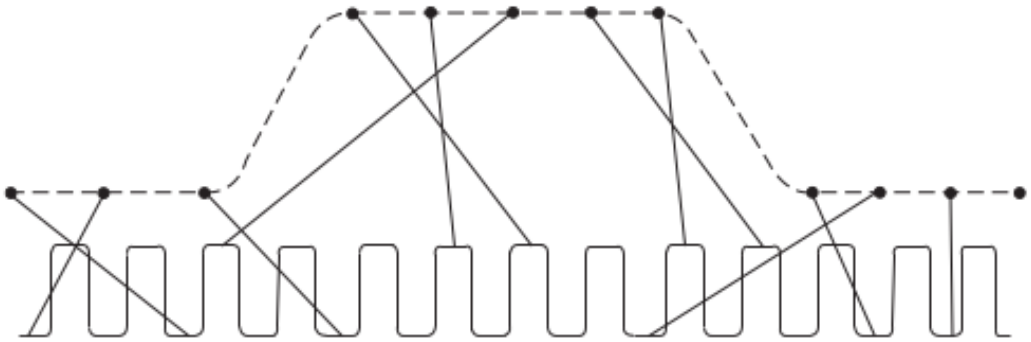
Slika 2.12: Diagram delovanja digitalnega fosfornega osciloskopa [31, str. 15]

Fosforna digitalna baza nam služi kot grafični pomnilnik, saj ima pomnilniško celico za vsako točko na zaslonu. Nahaja se po drugi stopnji, kjer iz zajetih točk naredimo valovno obliko (slika 2.12). Vsako novopridobljeno valovno obliko signala shranimo v bazo, tako da povečamo intenziteto točki, ki se je valovna oblika dotika. Iz baze se stanje periodično pošilja na zaslon. Zaradi paralelnosti se hitrost prikaza in zajema ne upočasni.

Vzorčni osciloskopi so podvrsta digitalnih osciloskopov, namenjena za zajem hitrih signalov, katerih frekvenca lahko doseže tudi do 90 GHz, kar je nekajkrat več od frekvence vzorčenja. To dosežemo s posebno vrsto vzorčenja, ki se imenuje `vzorčenje v ekvivalentnem času` (`equivalent time sampling`). Od vzorčenja v realnem času se razlikuje po tem, da vzorci niso zbrani v enkratnem prehodu signala, ampak so zbrani iz več ponovitev periode signala.

Vzorčenje v ekvivalentnem času [31, str. 26] je način vzorčenja, ki se uporablja za signale, katerih frekvenca je višja od 1/2 frekvence vzorčenja. Ker v eni periodi signala ne zajamemo dovolj vzorcev za pravilno rekonstrukcijo, uporabimo vzorce, ki jih zajamemo v več ponovitvah periode signala.

V vsaki ponovitvi periode signala zajamemo enega ali več vzorcev, iz katerih kasneje rekonstruiramo valovno obliko signala. Kot je razvidno iz slike 2.13, ni nujno, da so vzorci zajeti po vrsti, saj se pri rekonstrukciji signala vzorcem izračuna pravilen čas.



Slika 2.13: Vzorčenje v ekvivalentnem času [31, str. 26]

Ročni osciloskopi so podvrsta digitalnih osciloskopov, ki se od ostalih razlikuje po obliki. So majhni in prenosljivi, kar omogoča vgrajena baterija, zato so primerni za delo na terenu.

Računalniški osciloskopi so podvrsta digitalnih osciloskopov, ki nimajo vgrajenega zaslona. Za prikazovanje signala nam služi računalnik, osciloskop pa nanj priključimo preko vodila, kot je USB ali RS232. Osciloskop je po zgradbi enostavnejši, saj je dovolj, da vključuje samo vzorčenje. Obdelava vzorcev, kamor spada manjšanje števila točk in interpolacija, se lahko opravlja tudi na računalniku. Glavna prednost teh osciloskopov je njihova majhnost. Uporabimo jih v primeru, ko želimo signal analizirati s programom, ki se nahaja na osebnem računalniku.

Slika 2.14: Primer ročnega osciloskopa Metrix OX5022-C, s katerim lahko merimo napetost do 600 V. [14]

3. Uporabljene tehnologije

3.1 Strojna oprema

Pri izbiri strojne opreme so pomembni dejavniki:

- hiter `ADC`: hitrejši kot bo, hitrejše signale bomo lahko opazovali,
- dovolj pomnilnika: večji kot bo pomnilnik, daljši signal bomo lahko shranili vanj,
- podpora USB: za prenos vzorcev iz osciloskopa do računalnika,
- čim nižja cena,
- enostaven razvoj programske opreme.

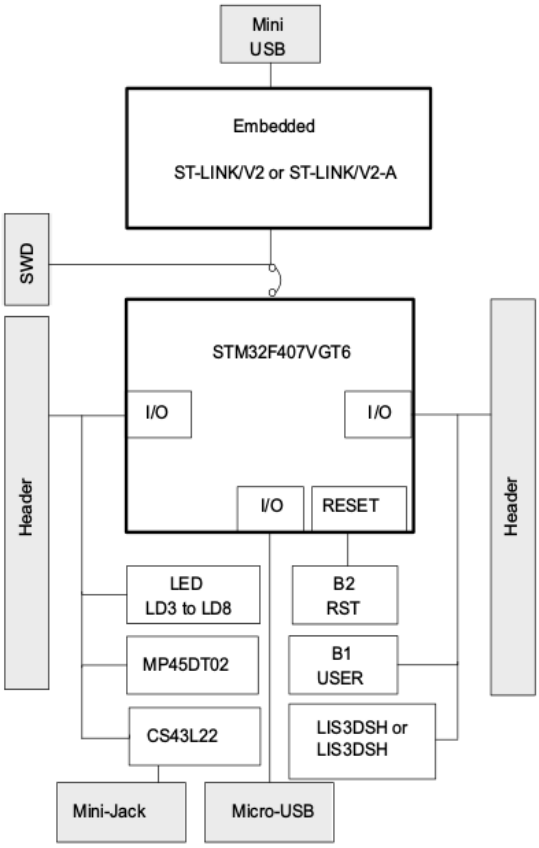
Izbrali smo razvojno ploščico `STM32F4-Discovery` z vgrajenim mikrokrmilnikom `STM32F407`, ki zadošča našim potrebam. Oboje proizvaja podjetje `STMicroelectronics` [21]. Dobre lastnosti te razvojne ploščice so: enostavno razhroščevanje in programiranje preko vmesnika `USB`, podpora prevajalnika `gcc` in nizka cena.

3.1.1 STM32F4-Discovery

STM32F4-Discovery je razvojna ploščica za mikrokrmilnik `STM32F407` [27]. Za nizko ceno (okoli 15 € [24]) nam ponuja:

- razhroščevanje in programiranje preko orodja `STLINK/v2`,
- vmesnik USB 2.0,

- napajanje ploščice preko vmesnika USB ali zunanjega napajanja 5 V,
- 3 V in 5 V zunanje napajanje,
- pospeškometer,
- mikrofon,
- 4 programabilne led diode,
- 2 gumba (uporabniški in "reset"),
- konektor za banana vtič, povezan na `DAC`.



Slika 3.1: Blok diagram razvojne ploščice STM32F4-Discovery [23, str. 9]

STLINK/v2 je orodje, ki ga je razvilo podjetje STMicroelectronics za programiranje mikrokrmilnikov STM [22]. Poleg programiranja omogoča tudi razhroščevanje.

STLINK/v2 je že vgrajen na razvojno ploščico (slika 3.2). Nanj se lahko povežemo preko vmesnika USB. STLINK/v2 se na mikrokrmilnik poveže preko vmesnika JTAG/SWD.

3.1.2 Mikrokrmilnik STM32F407

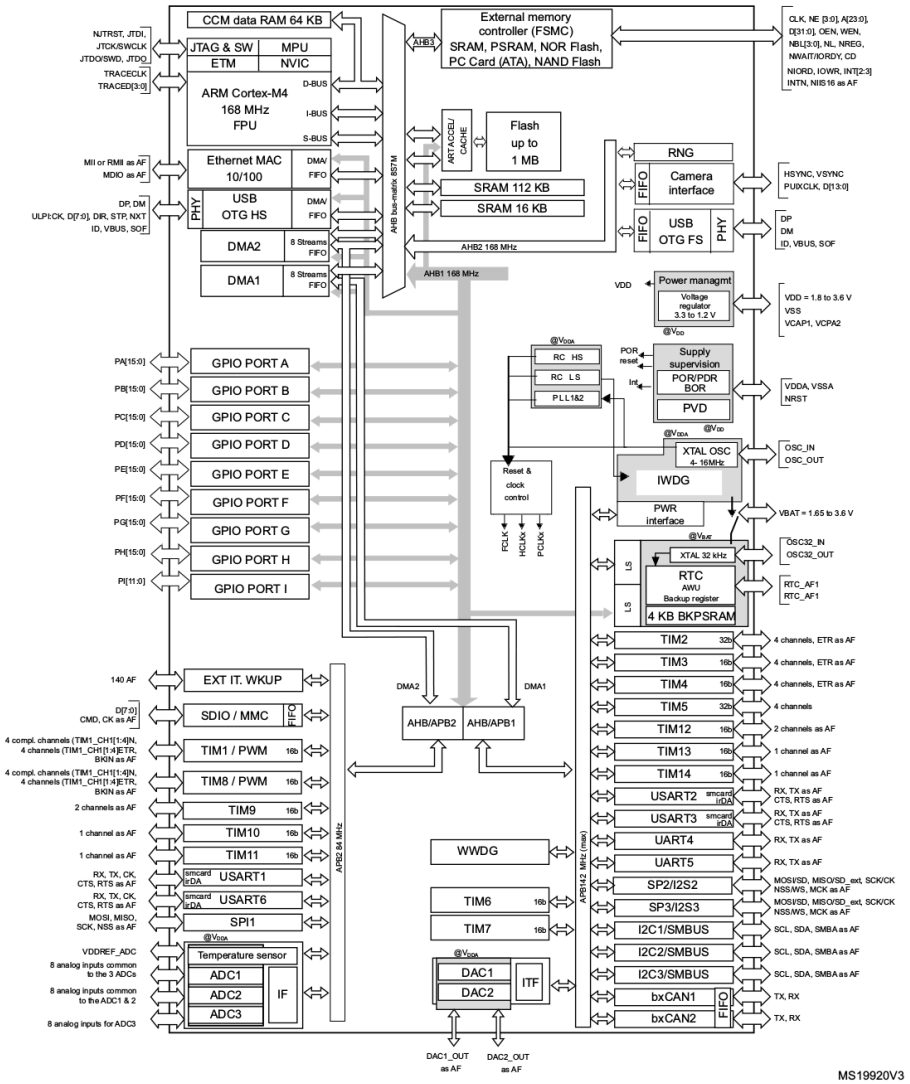
Mikrokrmilnik STM32F407 vsebuje 32-bitno jedro ARM Cortex M4 [8]. Njegova najvišja frekvenca je 168 MHz. Za program je namenjen 1 MB bralnega pomnilnika (flash memory). Za uporabo imamo na voljo 192 kB delovnega pomnilnika RAM, kar bo zadostilo našim potrebam. Iz tabele 3.1 je razvidno, kako dolg signal lahko shranimo v pomnilnik pri različnih nastavitvah ADC. Za referenco smo vzeli najhitrejše in najpočasnejše delovanje modula ADC. Ta čas je med 50 µs in 9 ms za 8-bitno ločljivost ter med 34 µs in 4,5 ms za 12-bitno ločljivost. Pri 12-bitni ločljivosti je dolžina signala za okoli faktor 2 manjša, saj en vzorec zaseda dva bajta.

Omeniti velja še, da so dolžine signala v praksi nekoliko krajše, saj del pomnilnika uporablja ostali del programa.

Ločljivost (b)	Δt (µs)	Št. vzorcev v pomnilniku	Dolžina signala (µs)
8	0,262	192000	50
12	0,357	96000	34
8	46,476	192000	8923
12	46,857	96000	4498

Tabela 3.1: Najdaljša dolžina signala, ki ga lahko shranimo v 192 kB pomnilnik.

Mikrokrmilnik vsebuje tri module ADC, ki lahko delujejo samostojno, ali pa jih povežemo skupaj in s tem pridobimo 3-krat višjo hitrost vzorčenja. Modul ADC lahko deluje s frekvenco 42 MHz, 21 MHz, 14 MHz ali 10,5 MHz. Lahko mu nastavimo tudi čas vzorčenja od 3 do 480 urinih ciklov [25, str. 423]. Čas vzorčenja



Slika 3.2: Blok diagram mikrokrmilnika `STM32F407` [26, str. 18]

izračunamo po enačbi (3.1) [25, str. 400]. Iz tabele 3.2 in 3.3 lahko razberemo čas vzorčenja pri različnih nastavitvah. V 8-bitnem načinu je frekvenca vzorčenja med 21,5 kHz in 3,8 MHz, v 12-bitnem načinu pa med 21,3 kHz in 2,8 MHz.

$$t = \frac{\text{št. bitov} + \text{št. ciklov vzorčenja}}{\text{frekvenca ADC}} \tag{3.1}$$

	Frekvenca modula ADC			
Čas vzorčenja (cikli)	**42,0 MHz**	**21,0 MHz**	**14,0 MHz**	**10,5 MHz**
3	0,262	0,524	0,786	1,048
15	0,548	1,095	1,643	2,190
28	0,857	1,714	2,571	3,429
56	1,524	3,048	4,571	6,095
84	2,190	4,381	6,571	8,762
112	2,857	5,714	8,571	11,429
144	3,619	7,238	10,857	14,476
480	11,619	23,238	34,857	46,476

Tabela 3.2: Časi vzorčenja, izraženi v mikrosekundah, veljajo za 8-bitne pretvorbe.

	Frekvenca modula ADC			
Čas vzorčenja (cikli)	**42,0 MHz**	**21,0 MHz**	**14,0 MHz**	**10,5 MHz**
3	0,357	0,714	1,071	1,429
15	0,643	1,286	1,929	2,571
28	0,952	1,905	2,857	3,810
56	1,619	3,238	4,857	6,476
84	2,286	4,571	6,857	9,143
112	2,952	5,905	8,857	11,810
144	3,714	7,429	11,143	14,857
480	11,714	23,429	35,143	46,857

Tabela 3.3: Časi vzorčenja, izraženi v mikrosekundah, veljajo za 12-bitne pretvorbe.

3.1.3 Pomanjkljivosti

Razvojna ploščica `STM32-Discovery` ima nekaj pomanjkljivosti, na katere bi lahko bili pozorni. Vendar bomo kljub temu lahko izdelali osciloskop.

Prva slabost je majhna notranja upornost modula `ADC` (R_{adc}), ki je vgrajen v mikrokrmilnik. Ta meri 6 kΩ [26, str. 132], kar pomeni, da ga bomo z merjenjem signala precej obremenili, rezultat pa bo nižja napetost od dejanske napetosti signala. Poleg tega osciloskop ne bo deloval z x10 sondo, saj so te umerjene na notranjo upornost 1 MΩ.

To pomanjkljivost bi lahko odpravili tako, da bi signal najprej speljali skozi ojačevalnik signala in ga nato signal pripeljali v mikrokrmilnik do modula `ADC`. Ojačevalnik signala ima lastnost, da ima visoko vhodno upornost in nizko izhodno upornost, zato bi manj obremenili izvor signala. Vendar to že presega obseg te diplomske naloge, zato se s tem ne bomo ukvarjali.

Druga slabost so referenčne napetosti U_{ref+} in U_{ref-}, ki so speljane v modul `ADC`. Te se uporabljajo za primerjanje z vhodno napetostjo, ki se mora nahajati med obema referenčnima napetostima. Glede na referenčne napetosti `ADC` določi napetost vhodnega signala. U_{ref-} je najnižja merljiva napetost, U_{ref+} pa najvišja.

Problem je, da na razvojni ploščici referenčne napetosti niso nastavljive, ampak so že določene. U_{ref+} je povezana na 3,0 V, U_{ref-} pa je vezana na 0 V, kar pomeni, da bomo z osciloskopom lahko merili le napetosti med 0 in 3 V.

Ta problem bi tudi lahko rešili z ojačevalnikom signala.

3.2 Programska oprema

Vsa programska oprema, ki smo jo uporabili, sodi pod odprtokodno programsko opremo.

Program za mikrokrmilnik `STM32F407` smo napisali v programskem jeziku C. Za prevajanje smo uporabili `GNU` skupino orodij za prevajanje `GCC ARM Embedded`, ki je prilagojena za arhitekturo `ARM`.

Program za mikrokrmilnik uporablja knjižnico `libopencm3`. To je knjižnica strojne programske opreme, ki podpira različne mikrokrmilnike tipa ARM Cortex M0(+)/M3/M4. Med drugim tudi mikrokrmilnik STM32F407. Strojna programska oprema zakrije uporabo registrov, s katerimi upravljamo posamezne enote

mikrokrmilnika, in na ta način poenostavi programiranje. Knjižnica podpira vse komponente mikrokrmilnika, ki jih bomo potrebovali. To so `GPIO` (general purpose input output), `ADC`, `DMA` (direct memory access) in `USB` (univesal serial bus).

Za razhroščevanje smo uporabili orodje `OpenOCD`, ki omogoča razhroščevanje s pomočjo izvorne kode. Na mikrokrmilnik se poveže posredno preko enote `STLINK/v2`. Poleg povezave na `STLINK/v2` ponuja še vrsto drugih povezav, ki pa jih ne bomo potrebovali. Ko smo enkrat povezani na mikrokrmilnik, ga lahko upravljamo z razhroščevalnikom `GDB`. Razhroščevalnik `GDB` omogoča:

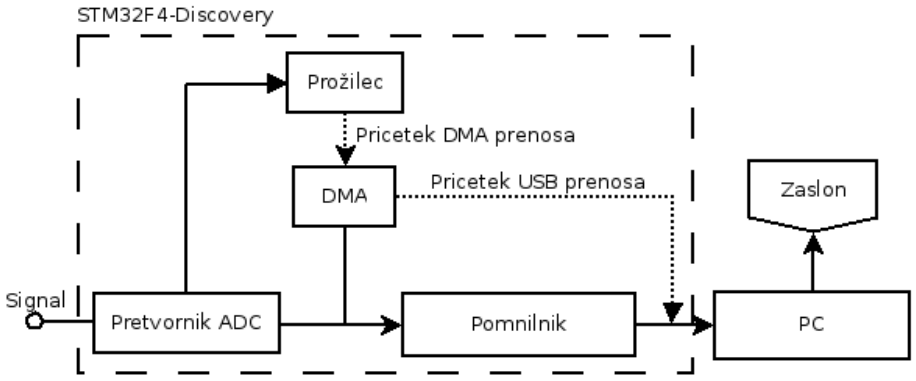
- vpogled v kodo, ki jo trenutno izvaja mikrokrmilnik,
- postavitev prekinitvenih točk,
- opazovanje spremenljivk v programu,
- opazovanje registrov,
- izvajanje programa po korakih,
- prekinjanje programa.

Uporabniški vmesnik na računalniku je v celoti napisan v programskem jeziku python.

4. Programska rešitev

Programska rešitev, ki smo jo razvili, je razdeljena na dva dela. Prvi program se izvaja na razvojni ploščici `STM32F4-Discovery` in je zadolžen za vzorčenje signala. Ker razvojni ploščici nismo dodali zaslona LCD, smo za prikaz uporabili zaslon osebnega računalnika, kjer se izvaja drugi program, ki zajete vzorce prikazuje. Osebni računalnik in razvojno ploščico med seboj povezuje vodilo USB.

Blok diagram osciloskopa je prikazan na sliki 4.1.



Slika 4.1: Blok diagram osciloskopa

4.1 Program na razvojni ploščici

Glavni enoti programa na razvojni ploščici sta pretvornik `ADC` in prožilec.

Pretvornik `ADC` nenehno pretvarja napetost v vzorce. Prožilec te vzorce spremlja, in kadar v njih zazna iskan dogodek, sproži zajem signala. To pomeni, da se vzorci začenjajo prenašati v pomnilnik s pomočjo vmesnika `DMA`.

4.1.1 Vrste dogodkov, ki jih lahko zazna prožilec

Prožilec lahko nastavimo na 5 vrst dogodkov.

Prvi način je poseben: brezpogojno takojšnje proženje. Ta način se uporablja za simulacijo delovanja osciloskopa brez prožilca. V tem primeru bo osciloskop nenehno zajemal sledi signala, kot da prožilca ne bi bilo.

Naslednja načina sta enostavna dogodka: proženje, kadar je signal v visokem ali v nizkem stanju. Pri tem načinu se prožilec sproži, kadar signal prekorači določen prag napetosti navzgor ali navzdol.

Zadnja dogodka sta sestavljena dogodka, saj ju sestavimo iz prej omenjenih enostavnih dogodkov. To sta prehod iz nizkega v visoko stanje in prehod iz visokega v nizko stanje.

Prehoda signala skozi napetost 0 V žal ne moremo zaznati, saj lahko merimo le napetosti med 0 V in 3 V. Več o tem v poglavju 3.1.3.

4.1.2 Delovanje prožilca

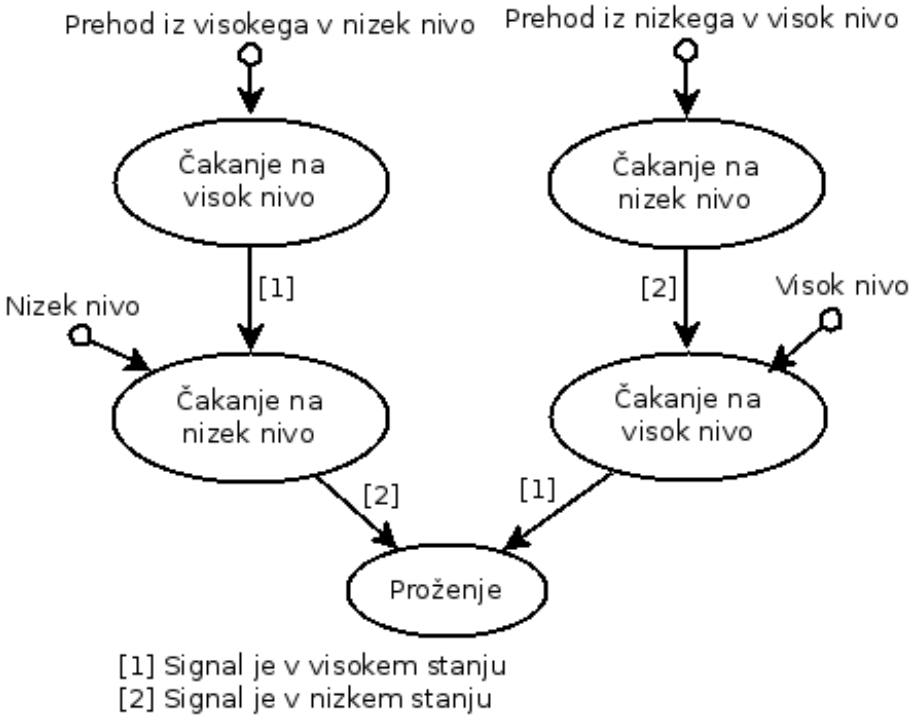
Pretvornik `ADC` ima v sebi enoto `AWD` (analog watchdog), ki lahko sproži prekinitev, kadar se pretvorjen vzorec nahaja izven določenih meja. To daje možnost, da na signalu zaznamo enostavne dogodke. V prekinitveni rutini se nahaja programska logika za zaznavo sestavljenih dogodkov. Logika prožilca je prikazana z Moorovim avtomatom stanj na sliki 4.2.

Brezpogojno proženje

Če bi radi brezpogojno sprožitev, nastavimo dovoljene meje modula `AWD` izven možnih vrednosti, ki jih lahko zasede vzorec. (Ker je vzorec največ 12-biten, register za meje pa 16-biten, to lahko storimo.) Tako se bo sprožilec sprožil ob vsakem zajetem vzorcu.

Proženje ob nivoju

Zaznamo lahko dva nivoja signala. Nizek nivo, ki je definiran z napetostjo med 0 V in 1,5 V, ter visok nivo, ki je definiran z napetostjo med 1,5 V in 3 V. Za zaznavo



Slika 4.2: Moorov diagram delovanja prožilca

nizkega ali visokega stanja na signalu je potrebno samo nastaviti pravilne dovoljene meje modulu `AWD`. Če bi naprimer radi proženje, kadar je signal v visokem stanju, nastavimo dovoljene meje na nizko napetostno območje. Enostavni dogodki so na diagramu 4.2 predstavljeni s povezavami med stanji.

Proženje ob prehodu

Prehod signala je sestavljen iz dveh enostavnih dogodkov, kar je prikazano tudi na diagramu 4.2. Za prehod iz nizkega v visoko stanje moramo najprej počakati, da signal pride v nizko stanje in šele nato v visoko stanje. Ista logika velja za prehod iz visokega v nizko stanje.

4.1.3 Ob proženju

Ko prožilec zazna na signalu iskan dogodek, se sproži. To pomeni, da pokliče programsko rutino, ki sproži `DMA` prenos iz registra `ADC` v pomnilnik. Število vzorcev, ki jih bo `DMA` prenesel v pomnilnik, določimo kot parameter vmesniku `DMA`. Za kolikšen čas bodo med seboj razmaknjeni zajeti vzorci, določimo s hitrostjo ure, ki poganja modul `ADC`, ter s časom, s katerim `ADC` vzorči vzorce.

4.1.4 Univerzalno zaporedno vodilo USB

`USB` (universal serial bus) je vodilo, namenjeno povezovanju različnih naprav na računalnik. V našem primeru bomo na računalnik povezali osciloskop.

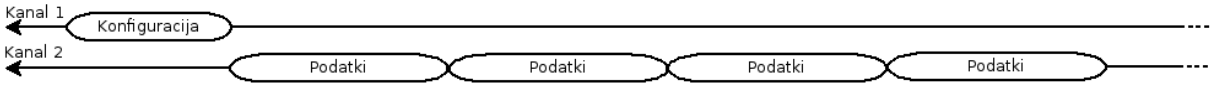
Prenos po vodilu `USB` je navidezno razdeljen na več kanalov, ki so oštevilčeni s končnimi številkami. Pri USB 2.0 FS (Full Speed) je največje število kanalov 32, pri čemer je 16 kanalov vhodnih in 16 izhodnih. Obstaja še en poseben oziroma kontrolni kanal, oštevilčen s številko 0. Ta kanal je dvosmeren in je namenjen krmiljenju naprave. Pod krmiljenje spada nastavljanje parametrov naprave ter pošiljanje ukazov.

Vmesnik USB nam med drugim omogoča tudi, da pri priklopu naprave na računalnik samodejno naloži potrebne gonilnike (ang: plug and play). V ta namen bi morali uporabiti že kakšen obstoječ vmesnik, ampak bomo zaradi lažjega programiranja uporabili lasten protokol in sprogramirali program za računalnik. Računalnik bo protokol prepoznal ter sprejel vzorce in jih prikazal.

Protokol za prenos valovne krivulje bo zajemal dva kanala. Najprej se bodo po prvem kanalu poslali parametri, s katerimi je bila zajeta valovna krivulja. Ti so:

- velikost vzorca v bitih,
- frekvenca ure,
- čas vzorčenja v periodah,
- število vzorcev.

Ko bo paket s parametri prispel na drugo stran, se bodo po drugem kanalu začeli prenašati vzorci, ki sestavljajo valovno krivuljo. Če se bo paket s parametri pri prenosu izgubil ali ne bo sprejet na drugi strani, se bodo vzorci zavrgli in osciloskop bo začel z zajemom novih.



Slika 4.3: Prenos vzorcev preko vodila USB

Slika 4.3 prikazuje prenos parametrov in vzorcev preko vodila USB. Ker je največja velikost paketa, poslanega preko vodila USB, 64 bajtov, je prenos vzorcev razdeljen na več paketov.

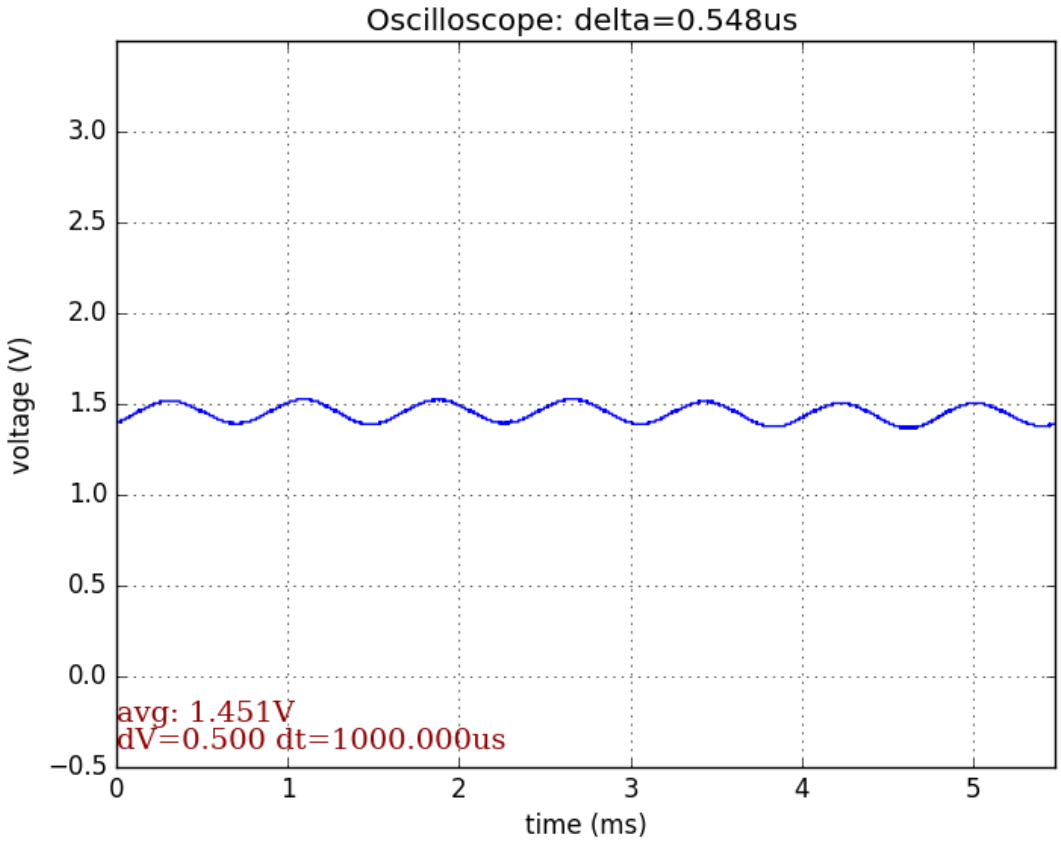
4.2 Program na osebnem računalniku

Program na osebnem računalniku je zadolžen za prikaz zajetih vzorcev. Poleg prikaza nam daje možnost, da sliko zamrznemo in jo podrobneje analiziramo, ali pa shranimo ter natisnemo.

Program je napisan v programskem jeziku python. Python smo izbrali zaradi njegove enostavnosti ter velike zbirke programskih knjižnic, ki so nam pomagale pri programiranju končne rešitve. V veliko pomoč je knjižnica matplotlib [7], ki služi izrisovanju grafa napetosti v odvisnosti od časa. Primer uporabniškega vmesnika z grafom vidimo na sliki 4.4.

4.2.1 Zagon programa

Program zaženemo iz konzolne vrstice in pri tem podamo dva parametra. Prvi parameter je dolžina sledi v vzorcih, drugi pa vrsta proženja. V primeru 4.1 smo osciloskop zagnali brez prožilca in določili, da valovna krivulja zajema 200 vzorcev. Druge možne vrednosti za prožilec so: `lth` (low to high) proženje ob prehodu z nizkega na visok nivo, `htl` (high to low) proženje ob prehodu z visokega na nizek nivo, `low` proženje ob nizkem nivoju ter `high` proženje ob visokem nivoju.



Slika 4.4: Grafični vmesnik osciloskopa

Koda 4.1: Zagon osciloskopa

```
./main.py -t disable -s 200
```

4.2.2 Uporabniški vmesnik

Ko program zaženemo, se nam na zaslonu prikaže uporabniški vmesnik, prikazan na sliki 4.4. Uporabniški vmesnik je sestavljen iz signala, prikazanega na grafu napetosti v odvisnosti od časa. Graf je z mrežo razdeljen na pravokotnike, kar omogoča lažje odčitavanje napetosti in časa iz grafa. V spodnjem levem kotu grafa je podan podatek, kolikšno napetost predstavlja višina pravokotnika in kolikšen

čas predstavlja dolžina pravokotnika. Poleg tega piše tudi, kolikšna je povprečna napetost prikazanega signala.

Graf s signalom se osveži vsakič, ko se zajame nova valovna krivulja. Sliko lahko zamrznemo s tipko za presledek (space). V tem primeru se slika ne bo več osveževala in nam bo ostala na razpolago za nadaljnjo analizo. Sliko odmrznemo tako, da ponovno pritisnemo tipko za presledek.

Do ukazov za upravljanje grafa dostopamo preko tipkovnice, do nekaterih pogostejših pa tudi preko orodne vrstice na dnu grafičnega vmesnika. Pogosteje uporabljeni ukazi so našteti v sledeči tabeli.

Tipka	**Pomen**	**Opis**
f	celozaslonski način	
p	premik in povečava	Z levim miškinim gumbom lahko signal premikamo, z desnim pa ga povečujemo in zmanjšujemo.
o	povečava na označeno območje	Graf signala povečamo na območje, ki ga označimo z miško.
h	ponastavitev pogleda	Ponastavimo povečavo in premik signala.
s	shranitev grafa	Graf signala shranimo v datoteko.
esc	izhod	

5. Meritve

Da bi osciloskop testirali, smo izmerili nekaj primerov iz vsakdanjega sveta. Prvi primer je pulzno-širinska modulacija, ki jo srečamo pri krmiljenju servomotorjev. Drugi primer je podatek, poslan po standardu RS232.

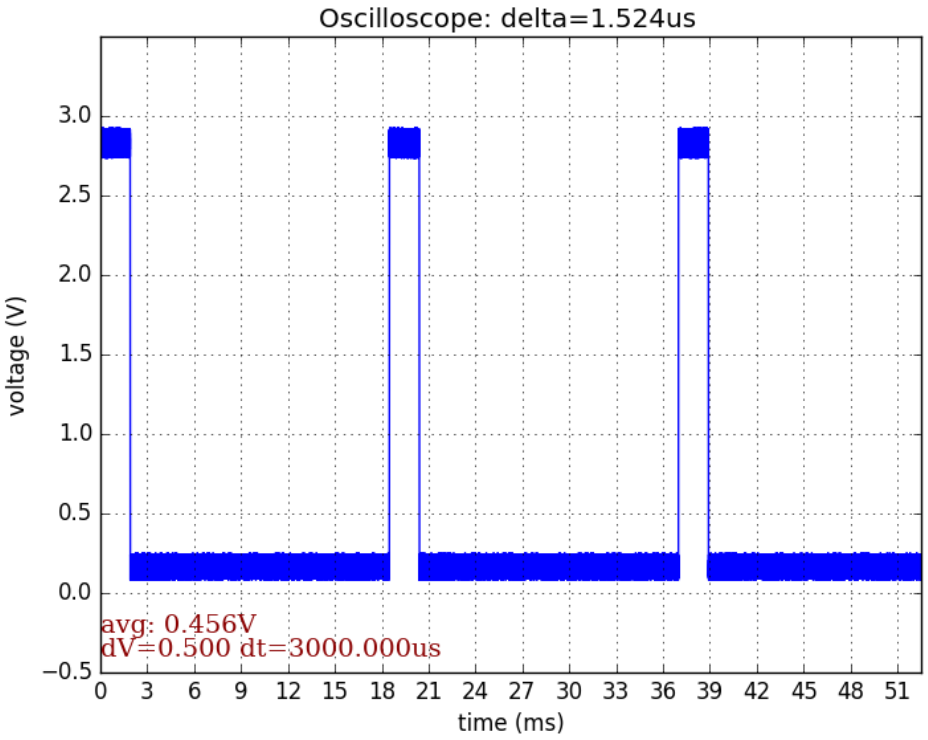
5.1 Servokrmiljenje

Krmiljenje servomotorjev se uporablja pri radijsko vodenih modelih. Signal, ki krmili servomotor, je moduliran s širino pulza, ki se ponovi vsakih 20 ms (lahko tudi bolj pogosto). Pulz signala je v stanju mirovanja širok okoli 1,5 ms, ko pa želimo premakniti servomotor za $\pm 90°$, se pulz skrajša/podaljša za okoli 0,5 ms.

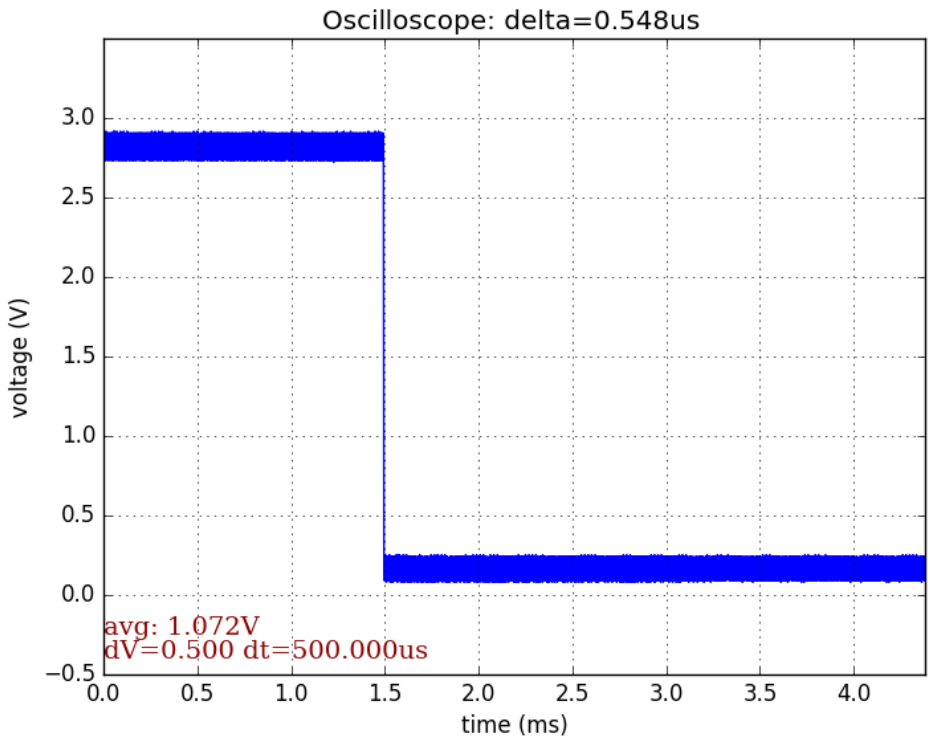
Pulz smo z osciloskopom zajeli tako, da smo nastavili prožilec, da se sproži, kadar gre signal iz nizkega stanja v visoko. S tem smo določili, da se prožilec sproži ob začetku pulza na signalu.

Sliki 5.2a in 5.2b prikazujeta pulz, ki smo ga zajeli s pomočjo osciloskopa. Pri $0°$ je širina pulza 1,5 ms, pri $-90°$ pa je širina pulza 1 ms, kar se ujema z modulacijo RV-signala.

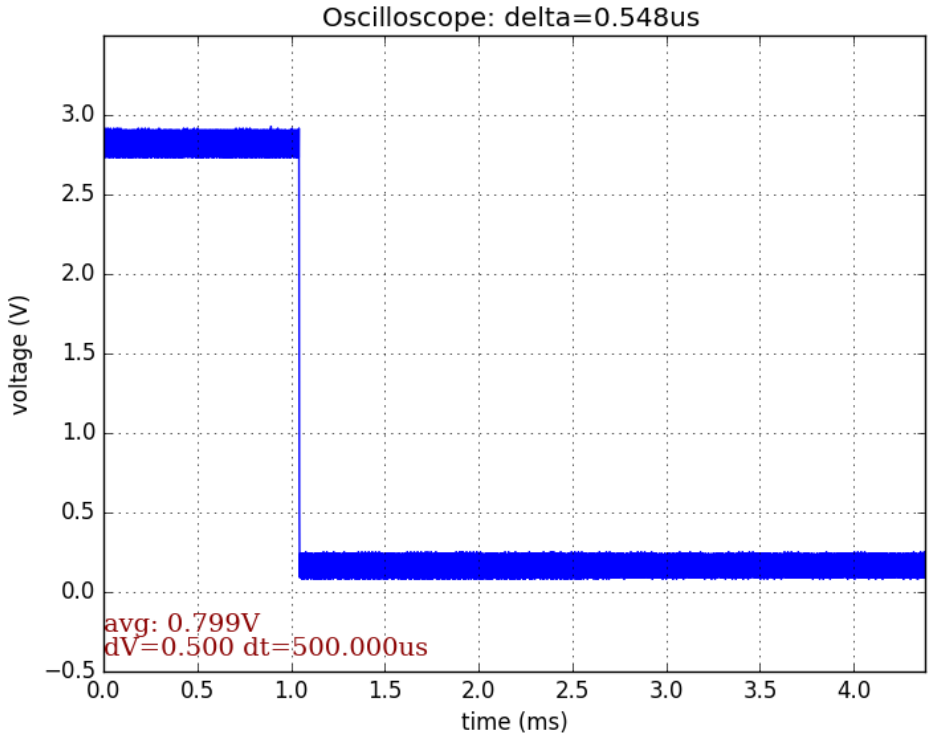
S slike 5.1 je razvidno, da se pulz ponovi vsakih 18 ms.



Slika 5.1: Pulzi signala PWM, ki si sledijo vsakih 18 ms



(a) Pulz signala PWM, ko je motor na 0°.



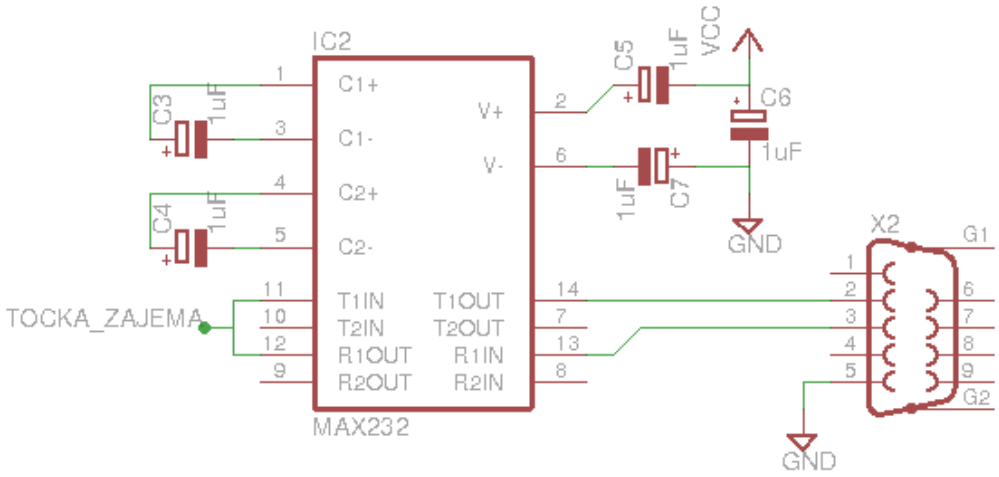
(b) Pulz signala PWM, ko je motor na −90°.

Slika 5.2: Primerjava PWM signalov

5.2 RS232

Da smo lahko z osciloskopom zajeli signal po standardu `RS232`, ki se prenaša preko vrat COM, smo morali najprej prilagoditi nivoje signala. Po standardu `RS232` je logična ničla predstavljena z napetostjo med 3 in 15 V, logična enica pa z napetostjo med –3 in –15 V. Pri pretvarjanju logičnih nivojev smo si pomagali s čipom `MAX232` [10], ki nivoje spremeni tako, da je logična ničla predstavljena z napetostjo 0 V in logična enica z napetostjo 5 V.

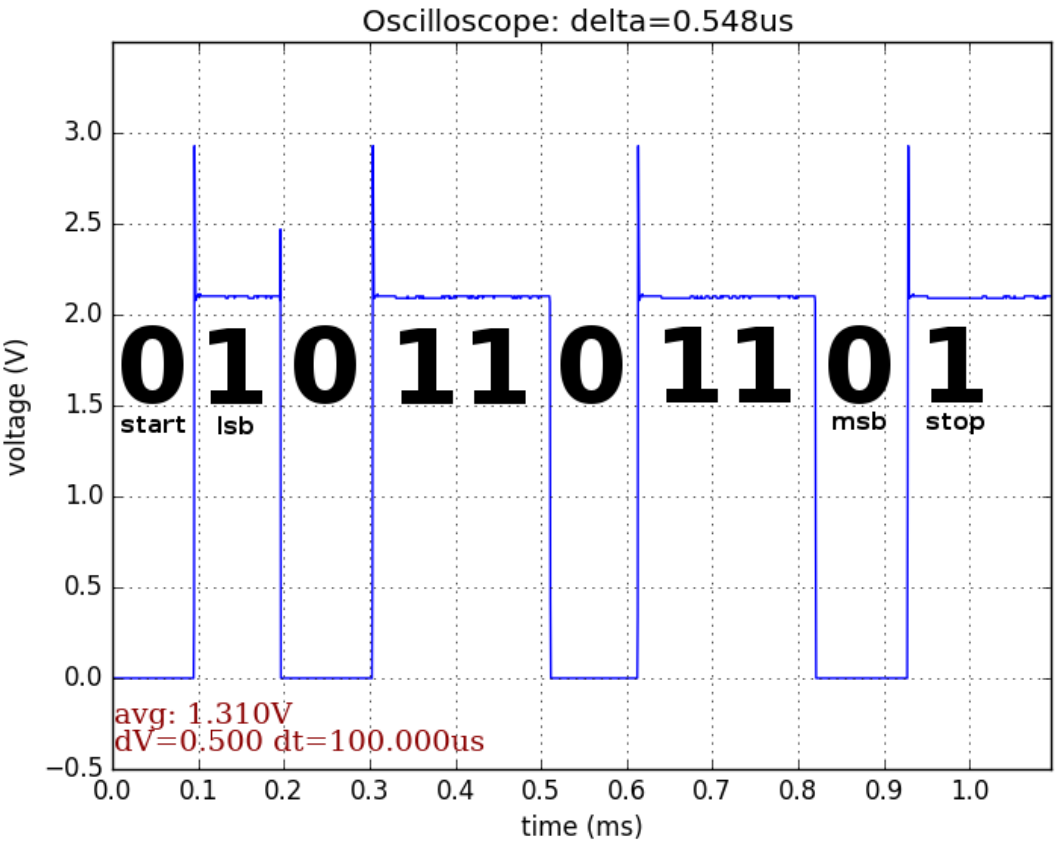
Na sliki 5.3 je prikazana shema, na kateri čip MAX232 pretvori logične nivoje.



Slika 5.3: Shema vezave za zajetje signala RS232

Ker je v stanju mirovanja na signalu logična enica, smo nastavili proženje prožilca, ko preide signal iz logične enice v ničlo.

Na sliki 5.4 je prikazan zajem signala RS232. Zraven signala so pripisani biti. Prvi bit je začetni bit, ki pove, da sledi prenos podatka. Sledi 8 bitov od `LSB` (least significant bit) do `MSB` (most significant bit) bita. Zadnji bit je stop bit, ki ponazarja konec prenosa. Če preberemo prenesene bite, dobimo binarno število 0b01101101 kar po desetiško 109 oziroma po tabeli ASCII predstavlja znak “m”, ki smo ga prenesli preko vrat COM.



Slika 5.4: Črka m, prenesena preko vrat COM po standardu RS232

6. Zaključek

V diplomski nalogi smo najprej opisali, kako osciloskop deluje. V drugem delu naloge smo razvojno ploščico STM32F4-Discovery, ki se dobi razmeroma poceni, uporabili kot digitalni osciloskop ter z njo zajeli signala PWM in RS232.

Ugotovili smo, da lahko signal zajamemo z vsakim mikrokrmilnikom, ki ima dovolj hiter pretvornik ADC, ter ga nato preko poljubne zaporedne povezave prenesemo na osebni računalnik, kjer ga prikažemo ali analiziramo. Videli smo, da izbira ploščice STM32F4-Discovery ni bila najboljša, saj ima pretvornik ADC referenčne napetosti že določene, kar pomeni, da lahko opazujemo le napetost med 0 V in 3 V, vendar nas v našem primeru to ni motilo.

Takšen enostaven osciloskop ima tudi druge pomanjkljivosti. Pri nekaterih meritvah bi potrebovali več vhodov, medtem ko ima naš osciloskop samo enega. Bolj kot to pa sem vsaj jaz pogrešal predproženje. To je mehanizem, ki poskrbi, da se dogodek, ki je sprožil zajem signala prikaže na sredini zaslona. V našem primeru se dogodek vedno prikaže na začetku zaslona, zato je dogajanje pred dogodkom očem skrito.

Tudi prikaz signala je bil zelo enostaven, saj smo zajete vzorce brez kakršnekoli obdelave prikazali s pomočjo knjižnice matplotlib. Če bi želeli kaj več, bi lahko uporabili že obstoječ program Xoscope [30], za zahtevnejše analize pa bi lahko uporabili program GNU Radio [4]. Seveda bi morali oba programa malo predelati, oziroma doprogramirati modul, ki podpira branje vzorcev iz našega osciloskopa.

Slike

Tabele

Literatura

[1] Diagram delovanja analognega osciloskopa. http://uenics.evansville.edu/~amr63/equipment/scope/graphics/oscillosco6_06.gif. Dostopano: 28. 4. 2016.

[2] Diagram delovanja digitalnega osciloskopa. http://www.tek.com/sites/tek.com/files/u811871/image%201%20July%20alan%20post.png. Dostopano: 30. 4. 2016.

[3] GCC ARM Embedded. https://launchpad.net/gcc-arm-embedded. Dostopano: 23. 3. 2016.

[4] GNU Radio. http://gnuradio.org/. Dostopano: 02. 8. 2016.

[5] Hires acquisition mode. http://www.tek.com/sites/tek.com/files/u811871/image%206%20july%20alan%20post.png. Dostopano: 30. 4. 2016.

[6] Knjižnica libopencm3. http://libopencm3.org/wiki/Main_Page. Dostopano: 23. 3. 2016.

[7] Matplotlib. http://matplotlib.org/. Dostopano: 22. 7. 2016.

[8] Mikrokrmilnik STM32F407VG. http://www.st.com/content/st_com/en/products/microcontrollers/stm32-32-bit-arm-cortex-mcus/stm32f4-series/stm32f407-417/stm32f407vg.html. Dostopano: 10. 3. 2016.

[9] Open On-Chip Debugger. http://openocd.org/. Dostopano: 23. 3. 2016.

[10] Opis čipa MAX232. http://datasheets.maximintegrated.com/en/ds/MAX220-MAX249.pdf. Dostopano: 22. 7. 2016.

[11] Oscilloscope Fundamentals. http://circuitslab.case.edu/manuals/Oscilloscope_Fundamentals_-_Tektronix.pdf. Dostopano: 1. 3. 2016.

[12] Peak acquisition mode. http://www.tek.com/sites/tek.com/files/u811871/image%205%20july%20alan%20post.png. Dostopano: 30. 4. 2016.

[13] Pulzno-širinsko moduliranje. https://en.wikipedia.org/wiki/Pulse-width_modulation. Dostopano: 14. 4. 2016.

[14] Ročni osciloskop Metrix OX5022-C. http://media.conrad.com/medias/global/ce/1000_1999/1000/1020/1027/102724_BB_01_FB.EPS_1000.jpg. Dostopano: 1. 3. 2016.

[15] RS232. https://en.wikipedia.org/wiki/RS-232. Dostopano: 14. 4. 2016.

[16] Sample acquisition mode. http://www.tek.com/sites/tek.com/files/u811871/image%204%20july%20alan%20post.png. Dostopano: 30. 4. 2016.

[17] Sample Processing in a Digital Oscilloscope. http://www.tek.com/blog/sample-processing-digital-oscilloscope. Dostopano: 29. 4. 2016.

[18] Servokrmiljenje. https://en.wikipedia.org/wiki/Servo_control. Dostopano: 14. 4. 2016.

[19] Slika osciloskopa. https://upload.wikimedia.org/wikipedia/sl/8/81/WTPC_Oscilloscope.jpg. Dostopano: 7. 2. 2016.

[20] Slika razvojne ploščice. http://www.st.com/st-web-ui/static/active/en/fragment/product_related/rpn_information/board_photo/stm32f4_discovery.jpg. Dostopano: 14. 3. 2016.

[21] Spletna stran podjetja STMicroelectronics. http://www.st.com. Dostopano: 14. 3. 2016.

[22] STLINK/v2 uporabniški priročnik. http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user_manual/DM00026748.pdf. Dostopano: 14. 3. 2016.

[23] STM32F4 discovery user manual. http://www.st.com/st-web-ui/static/active/cn/resource/technical/document/user_manual/DM00039084.pdf. Dostopano: 11. 5. 2016.

[24] STM32F4-Discovery v spletni trgovini. http://www.anglia-live.com/products/kw/stm32f4discovery/841569001_stm32f-discovery-eval-kit. Dostopano: 14. 3. 2016.

[25] STM32F4 reference manual. http://www.st.com/content/ccc/resource/technical/document/reference_manual/3d/6d/5a/66/b4/99/40/d4/DM00031020.pdf/files/DM00031020.pdf/jcr:content/translations/en.DM00031020.pdf. Dostopano: 21. 7. 2016.

[26] STM32F407 datasheet. http://www2.st.com/content/ccc/resource/technical/document/datasheet/ef/92/76/6d/bb/c2/4f/f7/DM00037051.pdf/files/DM00037051.pdf/jcr:content/translations/en.DM00037051.pdf. Dostopano: 11. 5. 2016.

[27] STM32F4DISCOVERY. http://www.st.com/web/catalog/tools/FM116/CL1620/SC959/SS1532/LN1848/PF252419. Dostopano: 10. 3. 2016.

[28] The GNU Project Debugger. https://www.gnu.org/software/gdb/. Dostopano: 23. 3. 2016.

[29] Tipi osciloskopov. https://en.wikipedia.org/wiki/Oscilloscope_types. Dostopano: 1. 3. 2016.

[30] xoscope. http://xoscope.sourceforge.net/. Dostopano: 02. 8. 2016.

[31] XYZs of oscilloscopes. https://engineering.purdue.edu/~aae520/tek-scope-primer.pdf. Dostopano: 25. 8. 2016.